An Extension of Atmospheric Boundary Layer Solvers to Include the PISO-Simple Algorithm

Adam Lavely Ganesh Vijayakumar James Brasseur

The Pennsylvania State University

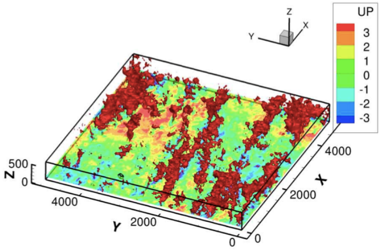
May 20, 2014

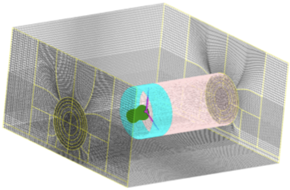
Contents

Contents

Our Research: Cyber Wind Facility

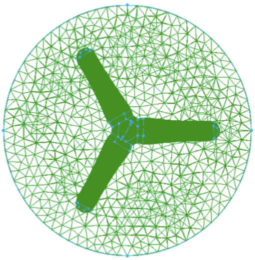
- Development of blade-resolved hybrid URANS-LES of the NREL 5 MW wind turbine
  - ◊ Vijayakumar et al.
- Compare to lower order models (BEMT, ALM)
  - ◊ Using the **same** inflow, algorithms, rotor settings, etc.

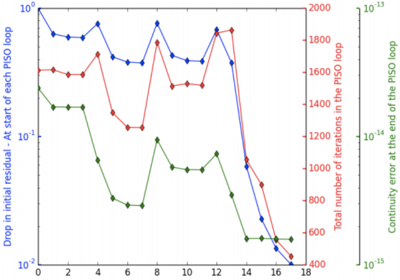




Our Research: Requirements

- Large time-steps
  - ◊ Courant and 'mesh Courant' numbers above 1
- Mesh rotation
  - ◊ Rotor rotates in cellZone with AMI
- Stability
  - ◊ Large changes in inflow, separation
- Minimal CPU time
  - ◊ Quick (and accurate) solutions





Transient Algorithm Overview

- SIMPLE:
  - ◊ Semi-Implicit Method for Pressure-Linked Equations
  - ◊ Spalding & Patankar at Imperial College circa 1970
  - ◊ simpleFoam - steady
  - ◊ transientSimpleFoam discontinued (but still online)

- PISO:
  - ◊ Pressure Implicit with Splitting of Operators
  - ◊ Issa at Imperial College circa 1986
  - ◊ pisoFoam

- PISO-SIMPLE:
  - ◊ Combination of PISO and SIMPLE
  - ◊ Unidentified origins
  - ◊ pimpleFoam, pimpleDyMFoam

PISO-Simple Code

From pimpleFoam.C:

```
while (runTime.run())
{
    #include "readTimeControls.H"
    ...
    runTime++;
     // --- Pressure-velocity PIMPLE corrector loop
     while (pimple.loop())
     {
        #include "UEqn.H"

        // --- Pressure corrector loop
        while (pimple.correct())
        {
            #include "pEqn.H"
        }
        if (pimple.turbCorr())
        {
            turbulence->correct();
        }
    }
    runTime.write();
    ...
}
```

From pEqn.H:

```
U = rAU*(UEqn() == sources(U))().H();
...
adjustPhi(phi, U, p);
 // Non-orthogonal pressure corrector loop
while (pimple.correctNonOrthogonal())
{
    // Pressure corrector
    fvScalarMatrix pEqn
    (
        fvm::laplacian(rAU, p) == fvc::div(phi)
    );

    pEqn.setReference(pRefCell, pRefValue);
    pEqn.solve(...);

    if (pimple.finalNonOrthogonalIter())
    {
        phi -= pEqn.flux();
    }
}
include "continuityErrs.H"
...
sources.correct(U);
```

- Time Loop
  - ◊ Outer Loop
    - ☐ Initial U equation
    - ☐ Initial p equation
    - ☐ Pressure Loop
      - ⋆ Non-ortho loop
        - ⇒ Update p
      - ⋆ Update U
    - ☐ Turbulence model

Number of Loops:

	PISO-Simple
Time	input
Outer	input
Pressure	input
Non-ortho	input

Input from fvSolution.

Algorithm Comparison

- Time Loop
  - ◊ Outer Loop
    - ☐ Initial U equation
    - ☐ Initial p equation
    - ☐ Pressure Loop
      - ⋆ Non-ortho loop
        - ⇒ Update p
      - ⋆ Update U
    - ☐ Turbulence model

Number of Loops:

	PISO-Simple	PISO	Transient Simple
Time	input	input	input
Outer	input	1	input
Pressure	input	input	1
Non-ortho	input	input	input

Input from fvSolution

Contents

- Time Loop
  - ◊ □ Wind turbine forces
    - □ Initial U equation
    - □ Initial p equation
    - □ Pressure Loop
      - ⋆ Non-ortho loop
        - ⇒ Update p
      - ⋆ Update U
    - □ Turbulence model

- U equation includes turbine forces
- Blade forces are given as values rather than matrix coefficients
- Relies on previous (not new) velocity field

PISO-Simple ALM Algorithm

- Time Loop
  - ◊ Outer Loop
    - ☐ Wind turbine forces
    - ☐ Initial U equation
    - ☐ Initial p equation
    - ☐ Pressure Loop
      - ⋆ Non-ortho loop
        - ⇒ Update p
      - ⋆ Update U
    - ☐ Turbulence model

- Wind turbine forces are more coupled with the velocity

- Outputs written out each time function is called

Write at last outer loop:

```
if (pimple.finalIter())
        {
            Write the output files
        }
```

Contents

Goals

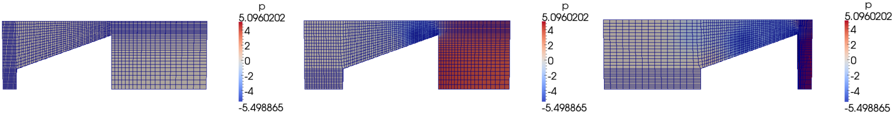
How to quickly/easily figure out how many loops to use?

$\rightarrow$ Desire **numerical stability** for **accurate simulation** with **minimum CPU time**

- Experiment on a simple test case
  - ◊ Use a small test that is 'representative' of our problem
  - ◊ Run many tests to identify broad characteristics
- Run tests for our application
  - ◊ See if characteristics found using test problem still apply
  - ◊ Small number of cases for small periods of time

Test case similarities:

- Moving/deforming mesh
- Varied cell volumes and aspect ratios
- Small mesh $\rightarrow$ runs quickly



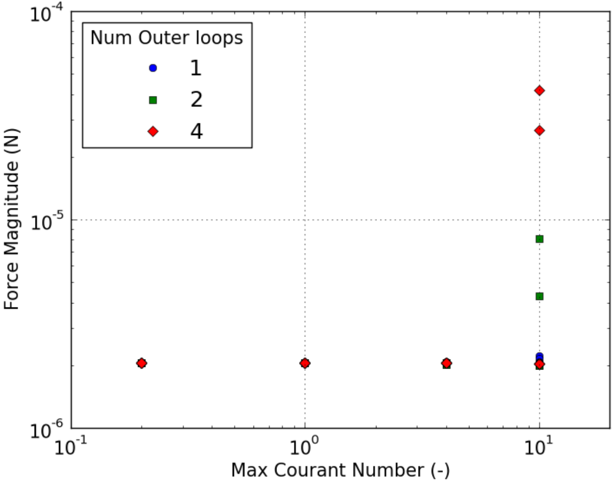
Controlled Inputs:

- *Number of outer loops = [1 2 4]*
- *Number of pressure loops = [1 2 4]*
- Number of non-ortho loops = [1 3]
- *Max Courant number = [.2 1 4 10]*

Outputs Compared:

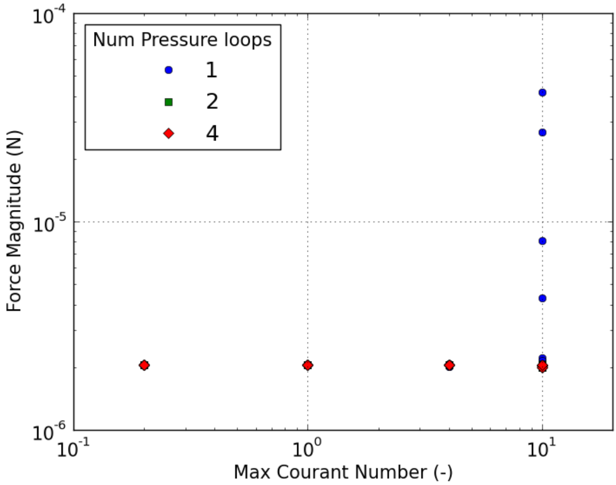
- Wall-clock time
- *Integrated force*
- Max pressure/velocity

Varying the Number of Outer Loops



Higher numbers of outer loops don't automatically provide more stability.

Varying the Number of Pressure Loops



Stability requires the predictor-corrector pressure loop to reach 'local equilibrium.'

Lessons Learned from Test Case

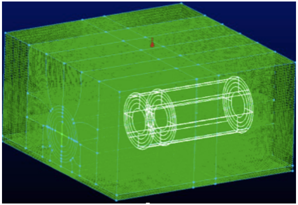
- Adding outer loops won't necessarily help accuracy
  - ◊ Predictor-corrector must converge, otherwise unstable
  - ◊ Non-ortho loops as required for mesh, regardless of pressure or outer loops
- Increasing the number of loops doesn't add time linearly
  - ◊ Tolerances met with fewer iterations in later loops
- Courant numbers above 1 easily reached
  - ◊ More outer loops required for longer time
  - ◊ Relies on pressure loop convergence
- Can reduce loops after initial transient
  - ◊ Initialization requires more loops than restart

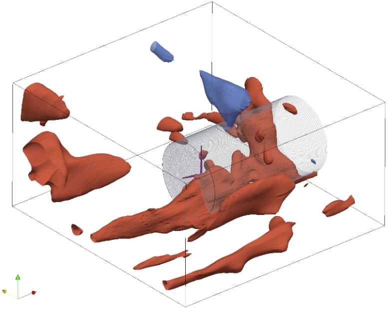
Application to Empty Domain ABL Solver

	ABL Inflow Initialization	ABL Inflow Run
Outer	3	2
Pressure	3	2
Non-ortho	5	3

PISO-Simple vs. PISO

- 13x the time-step
- 58% of the computational cost



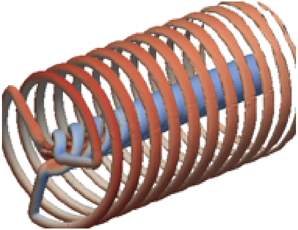


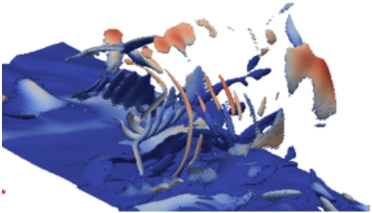
Application to Actuator Line Solver

	Uniform Inflow	ABL Inflow Initialization	ABL Inflow Run
Outer	2	3	2
Pressure	2	4	3
Non-ortho	3	5	3

PISO-Simple vs. PISO

- 10x the time-step
- 70% of the computational cost

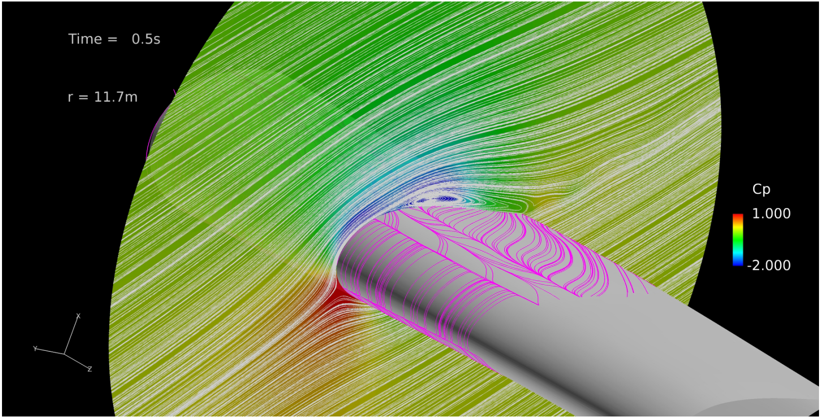




Application to Hybrid URANS-LES Simulations

	Initialization	Run
Outer	4	4
Pressure	4	2
Non-ortho	4	2

- Ganesh has also done work varying the number of pressure loop and non-ortho loops for each outer loop



Contents

Conclusions

- PISO-Simple is a combination of PISO and Simple
  - ◊ Coupled relaxation towards next time-step of Simple
  - ◊ Predictor-Corrector of PISO
- PISO-Simple allows for additional capabilities
  - ◊ Mesh motion and deformation
  - ◊ Larger time-steps with better stability
- Potential benefits are very application specific
  - ◊ Required for some applications
  - ◊ Adds unnecessary hassle to others

Acknowledgments

- **Funding:** PSU's ARL, NSF, DoE
- **ALM:** Pankaj Jha, Sven Schmitz, Matt Churchfield
- **OpenFOAM:** Brent Craven, Eric Paterson, Bryan Lewis
- **Pre-/Post- Processing:** Earl Duque, Warren Baker, Pointwise, William Brouwer
- **CPU:** XSEDE (NSF), RCC (PSU)

Thank you for your time.

Contact info: `adam.lavely@psu.edu`